AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A method for ~~protecting a byte code in a tracing framework~~ evaluating safety of a tracing program, comprising:

loading a byte code in a tracing framework, wherein the byte code comprises a plurality of instructions of the tracing program;

validating [[a]] the plurality of instructions when loading the byte code; [[and]]

performing at least one safety check on the plurality of instructions while ~~executing the plurality of instructions during a~~ performing virtual machine emulation of the plurality of instructions~~,~~

~~wherein the at least one safety check evaluates for a control transfer to an earlier instruction in the byte code sequence;~~

reporting an error condition and aborting virtual machine emulation of an unsafe instruction in the plurality of instructions when the at least one safety check detects the unsafe instruction; and

completing virtual machine emulation of a safe instruction in the plurality of instructions when the at least one safety check detects the safe instruction, after aborting virtual machine emulation of the unsafe instruction.

2-6. (Cancelled)

7. (Original) The method of claim 1, wherein the validating occurs during a single pass over the plurality of instructions.

8. (Original) The method of claim 1, wherein the byte code comprises an instruction set of a virtual machine.

9. (Original) The method of claim 8, wherein the instruction set comprises at least one selected from the group consisting of an arithmetic operation, a logical operation, a load operation, a store operation, and a control transfer operation.

10. (Currently Amended) The method of claim [[2]] 1, wherein the validating the plurality of instructions comprises:

verifying [[an]] that opcode bits for the plurality of instructions identify [[a]] valid operations~~; and~~

~~rejecting the tracing program if the opcode bit is invalid~~.

11. (Currently Amended) The method of claim [[2]] 1, wherein the validating the plurality of instructions comprises:

~~rejecting the tracing program if~~ evaluating whether an operand name referenced in the plurality of instructions does not refer to a valid operand provided by the virtual machine emulation.

12. (Currently Amended) The method of claim [[2]] 1, wherein the validating the plurality of instructions comprises:

computing a destination location within an instruction stream from one of the plurality of instructions; and

~~rejecting the tracing program if~~ evaluating whether the destination location is invalid.

13. (Currently Amended) The method of claim [[2]] 1, wherein the validating the plurality of instructions comprises:

determining whether a subroutine name is valid, when one of the plurality of instructions invokes a named subroutine~~; and~~

~~rejecting the tracing program if the subroutine is invalid~~.

14. (Currently Amended) The method of claim [[2]] 1, wherein the validating the plurality of instructions comprises:

~~summing~~ evaluating whether a total number of the plurality of instructions in the ~~input~~ byte code ~~stream; and~~

~~rejecting the tracing program if the total number~~ exceeds a user-configurable limit.

15. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating whether [[an]] the arithmetic operation results in a processor exception~~; and~~

~~aborting execution of a tracing program if any exception conditions result~~.

16. (Cancelled)

17. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the load operation before issuing [[the]] underlying instructions;

determining an appropriate alignment for the effective address; and

determining whether the effective address is improperly aligned based on the appropriate alignment

~~aborting execution of the tracing program if the appropriate alignment is improper~~.

18. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the store operation before issuing [[the]] underlying instructions;

determining an appropriate alignment for the effective address; and

determining whether the effective address is improperly aligned based on the appropriate alignment

~~aborting execution of the tracing program if the appropriate alignment is improper~~.

19. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of ~~the load or store~~ an operation for validity prior to executing the [[load]] operation, wherein the operation is one selected from the group consisting of the load operation and the store operation.

20. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating whether an effective address of ~~the load or store~~ an operation ~~against~~ falls outside a list of pre-computed address ranges assigned to a memory-mapped device hardware state, wherein the operation is one selected from the group consisting of the load operation and the store operation~~; and~~

~~aborting execution of the virtual machine emulation if the effective address falls within the list of pre-computed address ranges.~~

21. (Currently Amended) The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating whether an effective address of the store operation ~~against~~ falls outside a list of pre-computed address ranges assigned by the virtual machine to the tracing program~~; and~~

~~aborting execution of the virtual machine emulation if the effective address falls within the list of pre-computed address ranges.~~

22. (Currently Amended) A ~~mechanism for protecting a byte code,~~ tracing framework comprising:

an instruction validator configured to:

load a byte code comprising a plurality of instructions of a tracing program, and

validate [[a]] the plurality of instructions when loading the byte code; and

a safety check facility configured to:

perform at least one safety check on the plurality of instructions while ~~executing the plurality of instructions during a~~ performing virtual machine emulation of the plurality of instructions,

report an error condition and abort virtual machine emulation of an unsafe instruction in the plurality of instructions when the at least one safety check detects the unsafe instruction, and

complete virtual machine emulation of a safe instruction in the plurality of instructions when the at least one safety check detects the safe instruction, after aborting virtual machine emulation of the unsafe instruction~~,~~

~~wherein the at least one safety check evaluates for a transfer to an earlier instruction in the byte code sequence.~~

23. (Cancelled)

24. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate during a single pass over the plurality of instructions.

25. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the byte code comprises an instruction set of a virtual machine.

26. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction set comprises at least one selected from the group consisting of an arithmetic operation, a logical operation, a load operation, a store operation, and a control transfer operation.

27. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate [[a]] the plurality of instructions ~~comprises~~ by:

verifying [[a]] that opcode bits for the plurality of instructions ~~identifies a~~ identify valid operations~~; and~~

~~rejecting the tracing program if the opcode bit is invalid~~.

28. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate [[a]] the plurality of instructions ~~comprises~~ by:

~~rejecting the tracing program if~~ evaluating whether an operand name referenced in the plurality of instructions does not refer to a valid operand provided by the virtual machine emulation.

29. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate [[a]] the plurality of instructions ~~comprises~~ by:

computing a destination location within an instruction stream from one of the plurality of instructions; and

~~rejecting the tracing program if~~ evaluating whether the destination location is invalid.

30. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate [[a]] the plurality of instructions ~~comprises~~ by:

determining whether a subroutine name is valid, when one of the plurality of instructions invokes a named subroutine~~; and~~

~~rejecting the tracing program if the subroutine is invalid~~.

31. (Currently Amended) The ~~mechanism~~ tracing framework of claim 22, wherein the instruction validator is configured to validate [[a]] the plurality of instructions ~~comprises~~ by:

~~summing~~ evaluating whether a total number of the plurality of instructions in the ~~input~~ byte code ~~stream; and~~

~~rejecting the tracing program if the total number~~ exceeds a user-configurable limit.

32. (Currently Amended) The ~~mechanism~~ tracing framework of claim 26, wherein [[a]] the safety check facility is configured to perform the at least one safety check ~~comprises~~ by:

evaluating whether [[an]] the arithmetic operation results in a processor exception~~; and~~

~~aborting execution of a tracing program if any exception conditions result~~.

33. (Cancelled)

34. (Currently Amended) The ~~mechanism~~ tracing framework of claim 26, wherein [[a]] the safety check facility is configured to perform the at least one safety check ~~comprises~~ by:

evaluating an effective address of the load operation before issuing [[the]] underlying instructions;

determining an appropriate alignment for the effective address; and

determining whether the effective address is improperly aligned based on the appropriate alignment

~~aborting execution of the tracing program if the appropriate alignment is improper~~.

35. (Currently Amended) The ~~mechanism~~ tracing framework of claim 26, wherein [[a]] the safety check facility is configured to perform the at least one safety check ~~comprises~~ by:

evaluating an effective address of the store operation before issuing [[the]] underlying instructions;

determining an appropriate alignment for the effective address; and

determining whether the effective address is improperly aligned based on the appropriate alignment

~~aborting execution of the tracing program if the appropriate alignment is improper~~.

36. (Currently Amended) A computer system for ~~protecting a byte code in a tracing framework~~ evaluating safety of a tracing program, comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system to:

load a byte code in a tracing framework, wherein the byte code comprises a plurality of instructions of the tracing program;

validate [[a]] the plurality of instructions when loading the byte code; [[and]]

perform at least one safety check on the plurality of instructions while ~~executing the plurality of instructions during a~~ performing virtual machine emulation of the plurality of instructions; and

report an error condition and abort virtual machine emulation of an unsafe instruction in the plurality of instructions when the at least one safety check detects the unsafe instruction, and

complete virtual machine emulation of a safe instruction in the plurality of instructions when the at least one safety check detects the safe instruction, after aborting virtual machine emulation of the unsafe instruction~~,~~

~~wherein the at least one safety check evaluates for a control transfer to an earlier instruction in the byte code sequence~~.

37-41. (Cancelled)